High-Order In-Memory Hardware Accelerator for Combinatorial Optimization

D. Strukov[1*]

[1]*UC Santa Barbara, ECE Department, Santa Barbara, CA 93106, USA*

This talk presents recent work by my research group and collaborators in the DARPA QuICC program on developing hardware accelerators for combinatorial optimization problems [1–8]. The key innovations are twofold: (1) A massively parallel gradient computation method for high-degree polynomials [1,2], ideally suited for efficient mixed-signal in-crossbar-memory computing circuits (Fig. 1); and (2) a unified framework that applies this approach to design macro-blocks for solving higher-order problems, such as Boolean satisfiability (SAT) and polynomial unconstrained binary optimization (PUBO), directly in their native encoding [3].

Our gradient computation scheme accelerates gradient-descent-based heuristics, which are at the core of state-of-the-art local search SAT solvers and Ising machines. Its area complexity scales linearly with the number of variables and terms in the objective function polynomial, irrespective of its degree. In contrast, reformulating high-order problems into equivalent forms, such as the Quadratic Unconstrained Binary Optimization (QUBO) problem, incur high area overhead and exponentially expands the search space. This property, when coupled with the smoother energy landscapes characteristic of the native formulation [4], renders our approach particularly well-suited for solving high-order problems.

Experimental results from elementary tile prototypes, implemented using ReRAM and SRAM in-memory computing circuits [2,6,7] and benchmarked against small SAT instances, along with large-scale simulation studies [3], show significant improvements in speed and energy efficiency over existing solutions (Fig. 2). Finally, we introduce a family of scalable, higher-level architectures, called field-programmable Ising arrays (FPIAs), which integrate multiple tiles without using energy and performance taxing decomposition techniques [6,7] (Fig. 3). Inspired by island-style field-programmable gate arrays (FPGAs), the FPIA architecture leverages the sparsity of coupling matrices, common in many practical problems, to efficiently solve SAT problems with tens of thousands of variables on a single chip.

References

[1] M. Hizzani *et al*., "Memristor-based hardware and algorithms for higher-order Hopfield optimization solver outperforming quadratic Ising machines," in *Proc. ISCAS'24*, 2024, pp. 1–5.

[2] T. Bhattacharya *et al*., "Computing high-degree polynomial gradients in memory," *Nature communications*, vol. 15, art. 8211, 2024.

[3] T. Bhattacharya *et al*., "Unified framework for high-order heuristic solvers", submitted, 2025.

[4] D. Dobrynin *et al*., "Energy landscapes of combinatorial optimization in Ising machines", *Physics Review E*, vol. 110, art. 045308, 2024.

[5] T. Bhattacharya *et al*., "A fully integrated mixed-signal compute-in-memory accelerator for solving arbitrary order Boolean satisfiability problems," in *Proc. VLSISymp'25*, 2025, pp. 1-3.

[6] T. Bhattacharya *et al*., "KLIMA: Low-latency mixed-signal in-memory computing accelerator for solving arbitrary-order Boolean satisfiability", in *Proc. HotChips'25*, 2025, pp. 1-3.

[7] T. Bhattacharya, G.H. Hutchinson, G. Pedretti, D. Strukov, "HO-FPIA: High-order field-programmable Ising arrays with in-memory computing", in *Proc. ISVLSI'24*, 2024, pp. 252-259.

[8] G.H. Hutchinson, E. Sifferman, T. Bhattacharya, D.B. Strukov, "FPIA: Field-programmable Ising arrays with in-memory computing", in *Proc. ISLPED'24*, Sept. 2024, pp. 1-6.

Corresponding author: email: strukov@ece.ucsb.edu

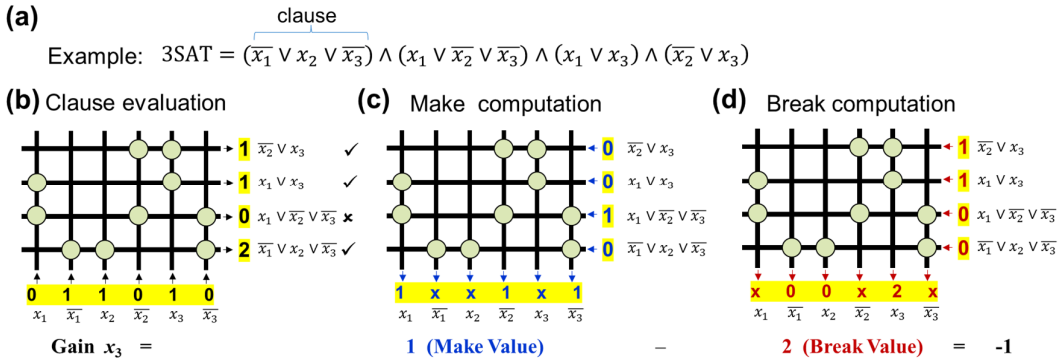

Figure 1. Massively parallel in-crossbar-memory computation of variable gains (i.e., partial derivatives of objective function in the context of SAT problems) for (a) a specific 3SAT instance and variable assignments [2]. The gain, defined as Gain ≡ Make – Break, represents the net increase in the number of satisfied clauses when flipping a given variable. (b) Clause evaluation is performed by applying digital voltages, corresponding to the current variable assignment, to the columns and sensing the row currents to identify unsatisfied clauses (output "0") and weakly satisfied clauses (output "1"). (c, d) Rows associated with unsatisfied / weakly satisfied clauses are then biased with digital "1" voltages, while grounding remaining ones, and the column currents are sensed to compute the (c) make / (d) break values. At the bottom of the panel, the gain for variable x_3 is illustrated as an example, though all variable gains are computed in parallel. The yellow background highlights the specific assignment and intermediate values at each step.

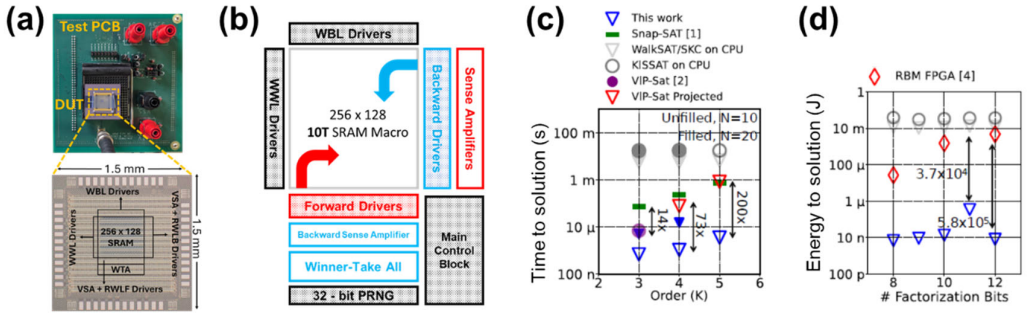

Figure 2. SRAM-based macro prototype for solving SAT problems [5]: (a) Chip micrograph and test board. (b) High-level block diagram showing a custom array of 10-transistor SRAM cells with peripheral circuitry, capable of running a local search SAT heuristic algorithm in a fully integrated mode. (c, d) Comparisons with other implementations of: (c) experimentally measured performance on 4-SAT hard random uniform benchmarks and (d) simulated energy consumption for semi-prime factoring problems using experimentally calibrated models.

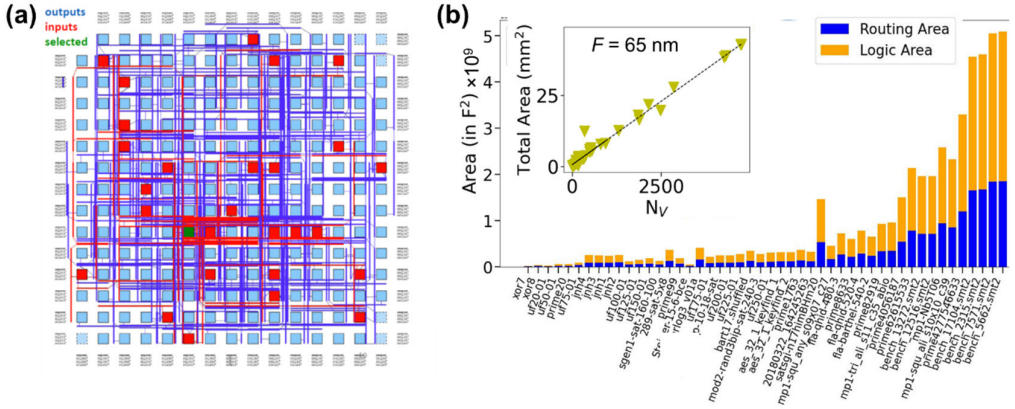

Figure 3. (a) Example mapping of the uf250-01.cnf problem from the SATLIB benchmark suite to a quadratic FPIA consisting of 15 × 15 tiles. Each tile hosts a macroblock shown in Figure 2 and is interconnected via an FPGA-like routing fabric [8]. The problem is first converted into an equivalent sequential circuit, after which the open-source Versatile Place and Route (VPR) tool is used for clustering, mapping, and detailed routing. For illustration, the panel highlights the inputs (red) and outputs (blue) of one randomly selected tile (green). (b) Simulated area for SAT2020 benchmark problems mapped onto the SRAM-based high-order FPIA, with a breakdown between the routing fabric and logic tiles. The inset shows projected area for FPIA implementation in 65-nm process node as a function of the number of variables for the same benchmark set.